Problem 1 Quicksort

In class, you saw a randomized incremental algorithm for point location in a trapezoidal decomposition of the plane. In this problem, we will look at a one-dimensional variant of the problem. Let \( P = \{x_1, x_2, \ldots, x_n\} \) be a set of \( n \) numbers, given in no particular order.

Consider the algorithm that picks a random permutation of \( P \) and then inserts the elements of \( P \) in this order into an (unbalanced) binary search tree \( T \).

(a) Explain how this algorithm can be interpreted as a one-dimensional version of the algorithm in class.

(b) Show that the expected running time for the construction of \( T \) is \( O(n \log n) \).

(c) Let \( z \) be a fixed number. Give a bound on the expected time it takes to search for \( z \) in \( T \). Here, the expectation is over the random permutation used to construct \( T \).

(d) Explain how this algorithm resembles randomized quicksort.

Problem 2 Trapezoidal map and search structure

Give an example of set of \( n \) line segments with an order on them that makes the algorithm we have seen in class create a search structure of size \( \Theta(n^2) \) and worst-case query time \( \Theta(n) \).

Problem 3 Point in polygon

Let \( P \) be a polygon given as an array of its \( n \) vertices in sorted order along the boundary. Give an algorithm that, given a query point \( q \), decides whether \( q \) lies inside \( P \) in \( O(\log n) \) time for the case where \( P \) is

(a) convex;

(b) \( y \)-monotone;

(c) star-shaped. Here you can assume that a witness point \( p \) in the interior of \( P \) that ‘sees’ every point in \( P \) is also given.